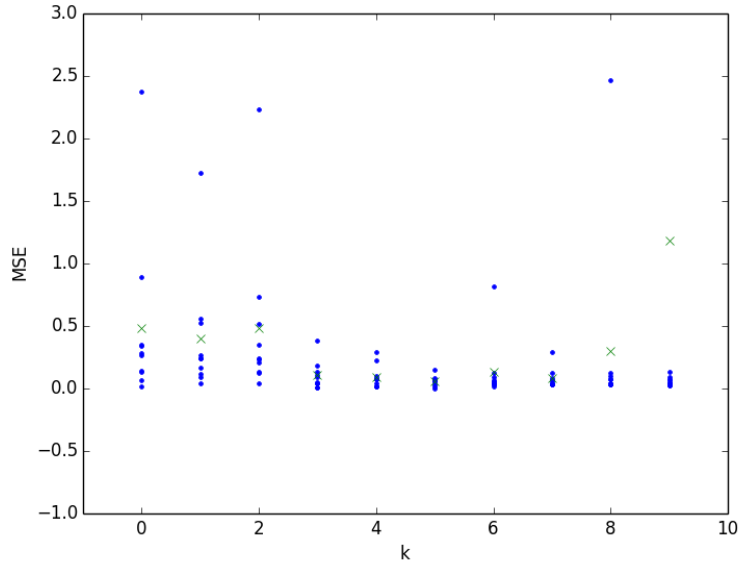
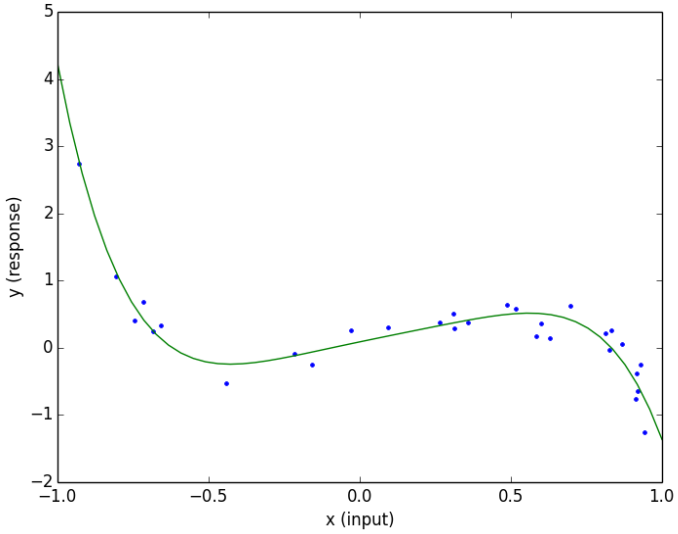
Greg Chan

EECS 349 Homework 2

Problem 1
Part A
See “nfoldpolyfit.py”



Above is the first output graph shown after running nfoldpolyfit with a max degree of 9, number of folds 10, and verbose output. That k = 3 through k = 5 are very good fits as the mean square error for these k values are very near zero, with only a few reaching as high as 0.5. Despite this however, k = 5 might still be a bit of an overfit of the data depending on what the data would like after more sampling.



This next graph shows the best fit line of a polynomial of degree k = 5. It seems to fit the data quite well and it has a low mean square error.

Lowest MSE: 0
Coefficients of the polynomial with the lowest MSE:
[-3.79952466 1.50277683 0.05905005 -0.1569019 0.94295249 0.08829073]
Degree of the polynomial with the lowest MSE: 5

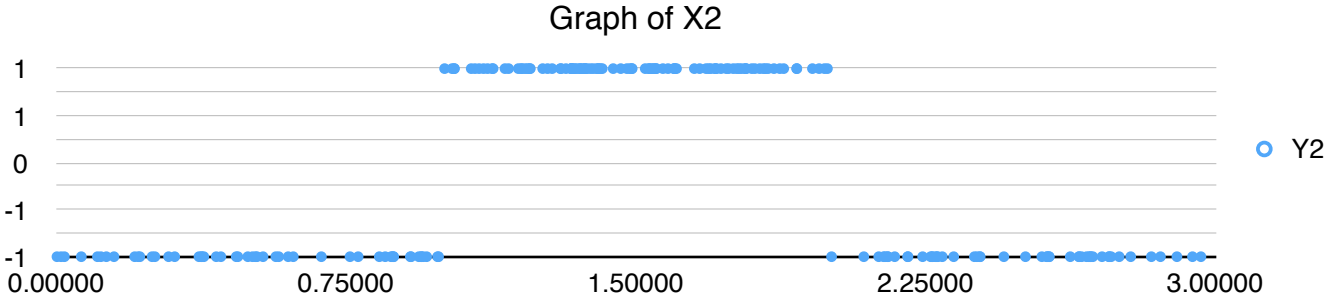
Part B
When k = 5, yielded the best mean square error result after running the k-fold cross-validation polynomial fit. This yielded the best results because it fit the data closest resulting in a low mean square error. Mean square error comprises of two parts, bias and variance. The bias portion of mean square error is the distance from the function average. Variance is how much the function varies over an interval. This function does not seemingly over fit the data, although it would be interesting to compare it against k = 3. What makes k = 5 better than k = 3 from a bias perspective is that the function is on average closer to the function than k = 3. Despite this, the data with values near 1 varies quite a bit and it would be interesting how the result might change if more data is collected.

Part C
Given the training data and the polynomial of degree k = 5, a query of x = 3 would likely be very negative, around maybe -750. This data would fit the model we have contracted, but given the training data's range, I don't think this model would be useful or accurate prediction because no other data ranges that far from zero. The values of the training data in the x dimension range from -1 to 1, which makes me believe that even values less than -1 or greater than 1 might be outside the searchable range for this data set for this model. In order to prove that our prediction for x = 3 is valid, we would need to collect more data ranging outside -1 and 1.

Problem 2
Part A
See "perceptrona.py".

The perceptron code returns a vector w = <4.01152, -11> and e = 6 for the vector X1 and Y1 as classifier.

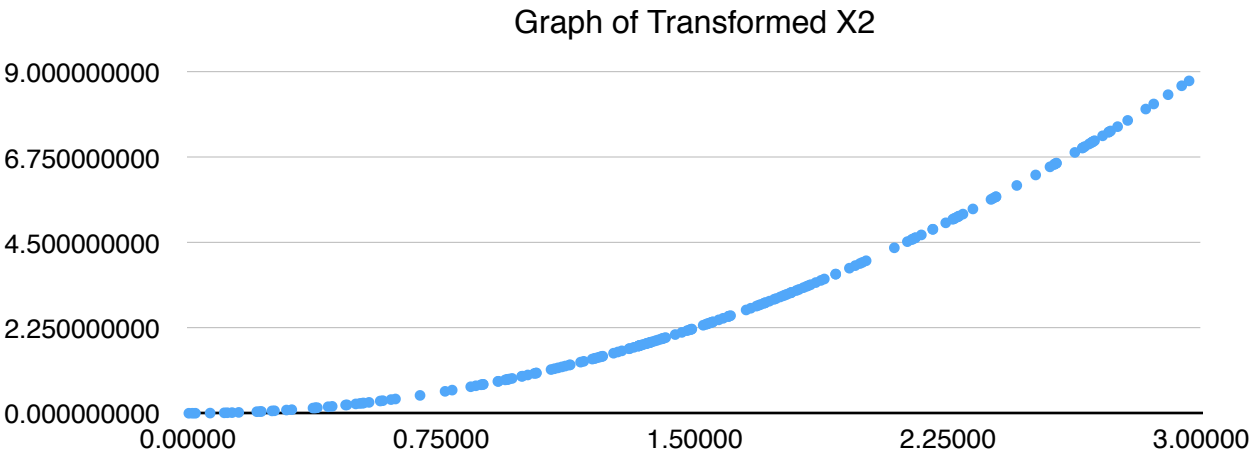
Part B
After running the program for a few minutes, it seemed evident that the perceptron algorithm would never stop because the vector X2 does not converge like X1. Convergence for this algorithm is contingent on the examples being properly classified, but without a transformation, it is impossible to achieve this with the vector X2. Plotting the data X2 against its classified data, it is apparent that it is not linearly separable, as seen below.



Part C

After graphing the data, it is apparent that this data cannot be separated linearly without a transformation (as stated in the problem statement). Thus, in order to make the data linearly separable, I transformed the data for X2 by adding a squared term to the vector. I modified my "perceptronc.py" by adding another weight to the weight vector and added a square term to X2 before running the same perceptron code from part A. Below are two graphs, which shows why squaring the terms in X2 produces a linearly separable set.

The axes of this chart are X2 and X2 squared. With this curve in the data, it is easy to draw a straight line through a portion of the curve to separate the data.

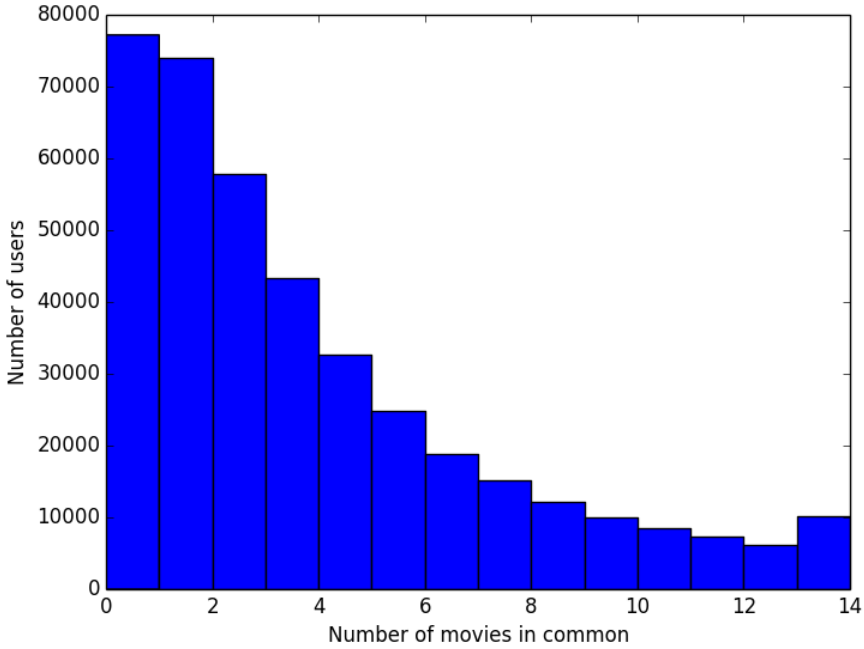


When the data it transformed with an added dimension for an X2 squared term, the program returns w = <73.3556525, -24.44140367, -49> and e = 685.

Problem 3

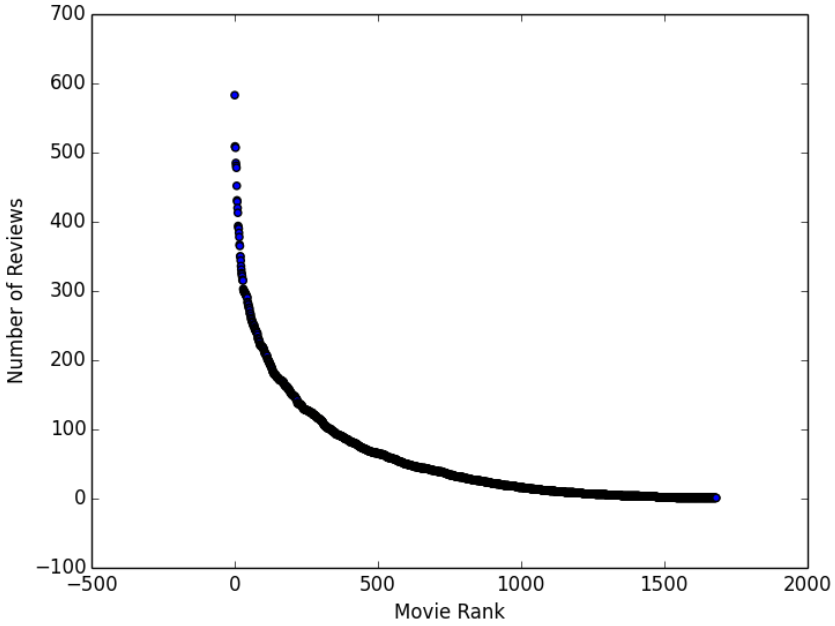
See "question3.py". There are three functions of importance in this file. The main function which runs the analysis for question 3 part a and b, the question 3a and question3b. question3a returns a histogram of the number of movies reviewed in common with the number of people who had those in common. This code outputs the mean number of movies two people have reviewed in common and the median. question3b returns the most reviewed movie and it's reviews as well as the least reviewed movie and its reviews as well as plotting a graph of the number of reviews each in order by the number of reviews.

Part A



Mean number of movies watched in common and rated the same: 5.728150
Median number of movies watched in common and rated the same: 3

Part B



Movie with most reviews is 50 with 583 reviews.
Movie with least reviews is 1682 with 1 reviews.

From the definition of Zipf's law on Wikipedia, it seems like this data does follow Zipf's law because the frequency of a movie's rating is inversely proportional to its rank, which follows the same logic as Zipf's law.

Problem 4
Part A
Given the example we covered in recitation, I think that the approach B, filling in the empty values with the average rating, is better. I think approach B is better than approach A because filling in a cell with zero guarantees that there will be some distance between two users for a movie if only one has rated it, which might not accurately reflect the actual distance when the rate the movies. That being said, having the values all be zero is useful in that if two users have not seen a movie, their rating of that movie does not count against them. Still, I think the averages are a better reflection of distance because those with similar tastes will have a well reflected value because it's based on what ratings they've given other movies in the past. For the example below, the averages both reflect what I think is a better value for distance between like minded ratings and disparate ratings.

Let's take Cathy and John, who have very similar ratings for Inception, Once and Star Wars. Although approach A says that these individuals are a distance of 2 apart, showing that they are close, approach B also has a similar distance of 2.5. However, for people with disparate tastes like Brian and Bob, approach A yields a distance between these two of 13, which is very high. Approach B, however I think yields a more realistic distance of 11.

Although I think that approach B is better than approach A, I think that the only way to determine if this is actually a useful metric is to test it with real world examples, and compare these predictions against those of the actual ratings once a user has actually watched the movie. This is however difficult.

Table with no modifications:

	Inception	Begin Again	Once	Star Wars	Avatar
Brian	2	?	3	?	4
Bob	5	1	1	5	2
Cathy	5	?	2	2	1
John	5	?	2	3	?

Approach A

	Inception	Begin Again	Once	Star Wars	Avatar
Brian	2	0	3	0	4
Bob	5	1	1	5	2
Cathy	5	0	2	2	1
John	5	0	2	3	0

Approach B

	Inception	Begin Again	Once	Star Wars	Avatar
Brian	2	3	3	3	4
Bob	5	1	1	5	2
Cathy	5	2.5	2	2	1
John	5	3.33	2	3	3.33

Part B

I think that Pearson's correlation distance would be preferable to Euclidean distance. It is a better measure of distance because I think it's likely that a user who has watched a movie like Inception will likely watch another Christopher Nolan movie like Memento, and what I think this means is that users will watch and enjoy movies of a similar genre and not watch movies of genres they don't enjoy. This means that these unrated movies won't impact the distance measure as much as they would for a Euclidean distance. Euclidean distance is heavily impacted by

Additionally, I think if the data is sparsely populated and many users haven't watched a good portion of the movies, Pearson's correlation will result in correlated data because of all the placeholder values, which may on average produce a better movie recommendation. Also based on the average movies that users have watched in common from the earlier problem, with an average of approximately 18 views in common, it is likely that there will be little data to base Euclidean distance off of.

	Inception	Begin Again	Once	Star Wars	Avatar
Brian	2	3	3	3	4
Bob	5	1	1	5	2
Cathy	5	3	2	2	1
John	5	3	2	3	3

Using Euclidean distance, we can calculate the distance between Avatar and all the other movies.

(Avatar, Inception) = 5.7
(Avatar, Begin Again) = 2.44
(Avatar, Once) = 2
(Avatar, Star Wars) = 3.3

Using Pearson's correlation:

(Avatar, Inception) = -0.94
(Avatar, Begin Again) = 0.188
(Avatar, Once) = 0.6546
(Avatar, Star Wars) = 0.142

With missing ratings filled in with the value 3, it seems that both the Pearson and Euclidean distance predict that viewers of Once and Avatar will rate them similarly. I'm not surprised that both of these performed similarly, but with 943 users and 1682 movies, I think it'll be much more likely for Pearson's correlation to predict ratings better than Euclidean distance.

Problem 5
See "user_cf.py"
See "item_cf.py"

Problem 6
I was unable to run the data 100 fold cross-validations, but Professor Pardo said we could do it on a smaller set of 15 or so. I did mine with 18.

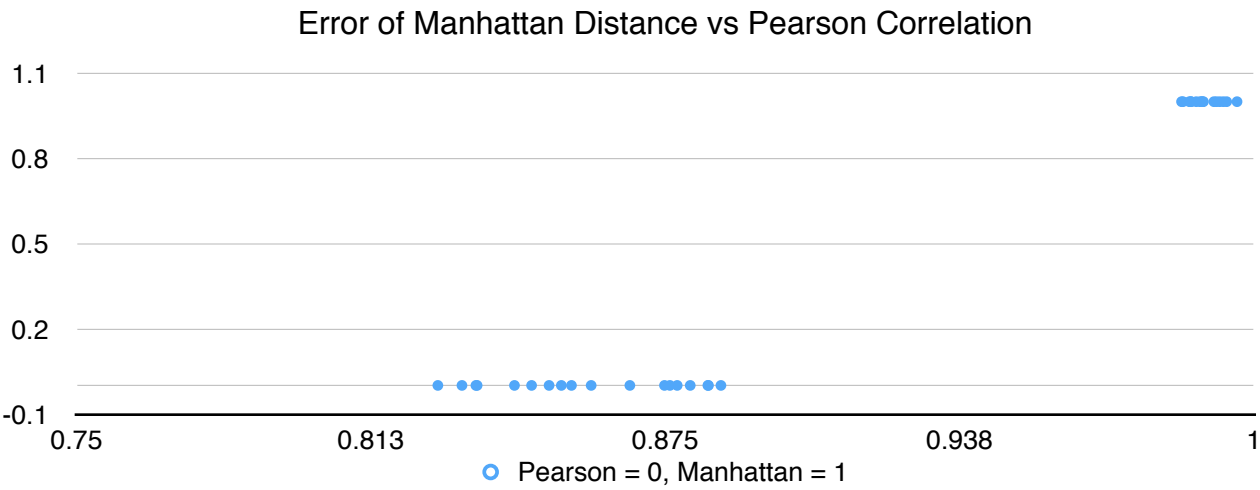
Part A
I decided to use the simple error percentage of (actual - predicted) / actual. I chose this because I think it will accurately depicted the percentage of error with which our collaborative filter when comparing two users tastes in movies. I also chose it because it's an easy to compute and simple to interpret measurement. It's positive if the prediction is over that of the actual rating and negative if it under predicted the rating. While it's not a perfect measurement of error, in that ratings with 0 will always yield a large error, I think it will demonstrate the impact that zero values will have on predicting good results for large amounts of data. I expect the majority of errors to be above 95%.

Part B
I decided to uses Welch's t test to determine if the difference between my two variant systems are statistically significant. I chose Welch's t test because it is used for unpaired or independent samples t tests, and it is good for when two samples are being compared are non-overlapping. This works well with the data I will be analyzing because the Manhattan and Pearson data for instance is not paired, but this t test will still help me determine if the difference between the two is statistically significant. It is also useful because unlike Student's t test, we do not know if the distribution of our data is normal. If the p-value for the data is less than 0.05, the difference in the data is statistically significant.

Part C
For part C, I tested Manhattan distance against Pearson correlation keeping k as a constant 100, with i as 1, with 15 folds. From the parameters, it appears that my intuition for distance is right as the average error for Manhattan distance appears to be approximately 99% on average. Where as Pearson correlation seemed to cope with the zeroed out values better yielding approximately an 86% error on average. While both of these are still very high, the Pearson correlation seems to outperform the Manhattan distance on average when k equals 100, i is 1 and only over 15 folds.

Manhattan K = 100, i = 1, 18 folds	Pearson's K = 100, i = 1, 18 folds
0.984616666667	0.867483333333
0.986633333333	0.850333333333
0.984916666667	0.826716666667
0.989266666667	0.874833333333
0.986266666667	0.831833333333
0.992733333333	0.855083333333
0.992033333333	0.852933333333
0.992166666667	0.83505
0.988533333333	0.8803
0.988683333333	0.88405
0.991483333333	0.886816666667
0.993483333333	0.8348
0.9942	0.846616666667
0.987733333333	0.884166666667
0.9964	0.859266666667
0.9889	0.842983333333
0.986616666667	0.875983333333
0.989033333333	0.877533333333

The two tailed p value of the Welch's t test resulted in a value less than 0.0001. This means that the difference between our two data sets is likely to be statistically significant. Based on just a glance at the data, it seem that this is supported given that the Pearson's worst run seemed to outperform the Manhattan run.



Part D
Keeping k = 100, using Pearson correlation distance.

For collaborative filter it makes sense for i to be zero because it will remove values of little meaning from the prediction process, and more meaningful values will be returned form the collaborative filter. It does in that I believed that filling in zeros would cause unnecessary error because it will impact the distance calculations by making them larger for Euclidean/Manhattan distances. Thus, Pearson correlation seems to work better when these values are excluded.

Pearson i = 0	**Pearson i = 1**
0.751516666667	0.867483333333
0.804583333333	0.850333333333
0.7641	0.826716666667
0.76785	0.874833333333
0.792233333333	0.831833333333
0.772816666667	0.855083333333
0.7893	0.852933333333
0.812066666667	0.83505
0.784383333333	0.8803
0.759683333333	0.88405
0.760166666667	0.886816666667
0.799633333333	0.8348
0.784216666667	0.846616666667
0.746033333333	0.884166666667
0.794683333333	0.859266666667
0.7823	0.842983333333
0.78115	0.875983333333
0.774666666667	0.877533333333

Part E
For this section of the assignment, I used Manhattan and Pearson correlation distance as the distance measure for filtering. This resulted in interesting results as Manhattan distance seems to not be a very good measurement between users. I was able to run it 18 fold times given the time I had.

Manhattan:

The trend from my data is that increasing k will result in a worse prediction. This likely because the tie breaker for selecting a predicted rating is the mode of the k nearest neighbors, which

may be more common as you get further from the original point of interest. It would be an interesting exercise to see the result of say the average predicted rating over the different k values as well.

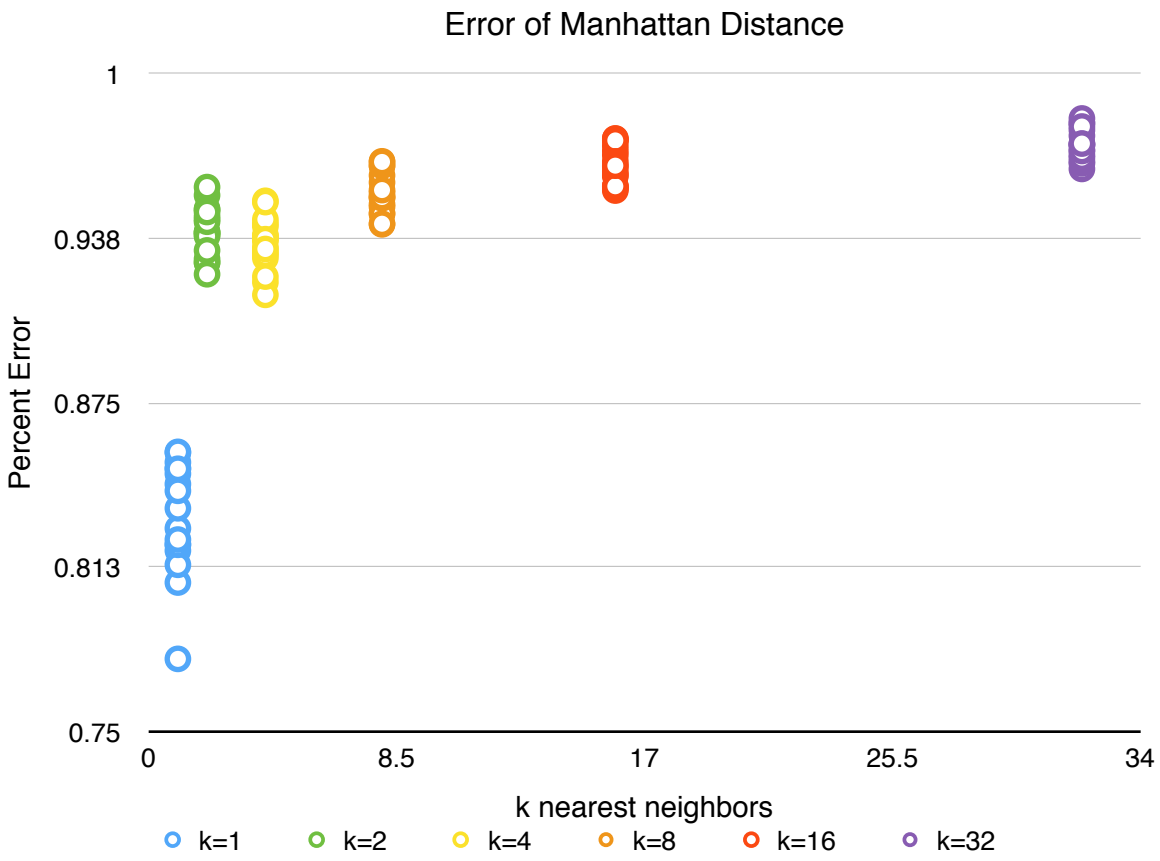
Below is a table of the different k values and the error from the actual value as a percentage represented in decimal form. As you can see, the difference between the values for each k is fairly consistent but the difference between k = 1 and k = 2 is large.

k = 1	k = 2	k = 4	k = 8	k = 16	k = 32
0.777683333333	0.928483333333	0.92095	0.9472	0.956366666667	0.964016666667
0.806716666667	0.929733333333	0.932166666667	0.94295	0.9558	0.966116666667
0.81885	0.940116666667	0.938383333333	0.9506	0.964933333333	0.96685
0.821266666667	0.9485	0.943283333333	0.949616666667	0.963	0.97665
0.813466666667	0.939733333333	0.936633333333	0.958883333333	0.969266666667	0.97295
0.8481	0.94595	0.9381	0.9587	0.9756	0.9788
0.827316666667	0.938933333333	0.9301	0.946833333333	0.961266666667	0.969283333333
0.852433333333	0.954	0.9519	0.96155	0.970866666667	0.981483333333
0.821066666667	0.9283	0.931433333333	0.95505	0.96175	0.968316666667
0.83495	0.944783333333	0.938883333333	0.943183333333	0.957416666667	0.968783333333
0.850183333333	0.957116666667	0.94215	0.9653	0.9748	0.98145
0.823066666667	0.923966666667	0.916133333333	0.9536	0.97275	0.97315
0.856116666667	0.94885	0.944833333333	0.966916666667	0.975516666667	0.983216666667
0.844166666667	0.939833333333	0.93415	0.95285	0.957233333333	0.971166666667
0.8416	0.93845	0.9373	0.965183333333	0.974866666667	0.98015
0.84165	0.933033333333	0.923083333333	0.953333333333	0.957433333333	0.973583333333
0.856416666667	0.943933333333	0.933566666667	0.966683333333	0.966883333333	0.97085
0.849983333333	0.947633333333	0.9514	0.955983333333	0.965266666667	0.973666666667

The two tailed p value of the Welch's t test resulted in a value less than 0.0001 when testing k = 1 and k = 2.

After performing the Welch's t test on k = 1 and k = 2, it is apparent that there is a statistical significant difference between k = 1 and k = 2. And it thus caries that k = 1 and k = 4, 8, 16, and 32 are all also statistically significant as the values for each subsequent value of k result in higher, worse error.

Below is a graph of the error of the Manhattan distance as a function of k.

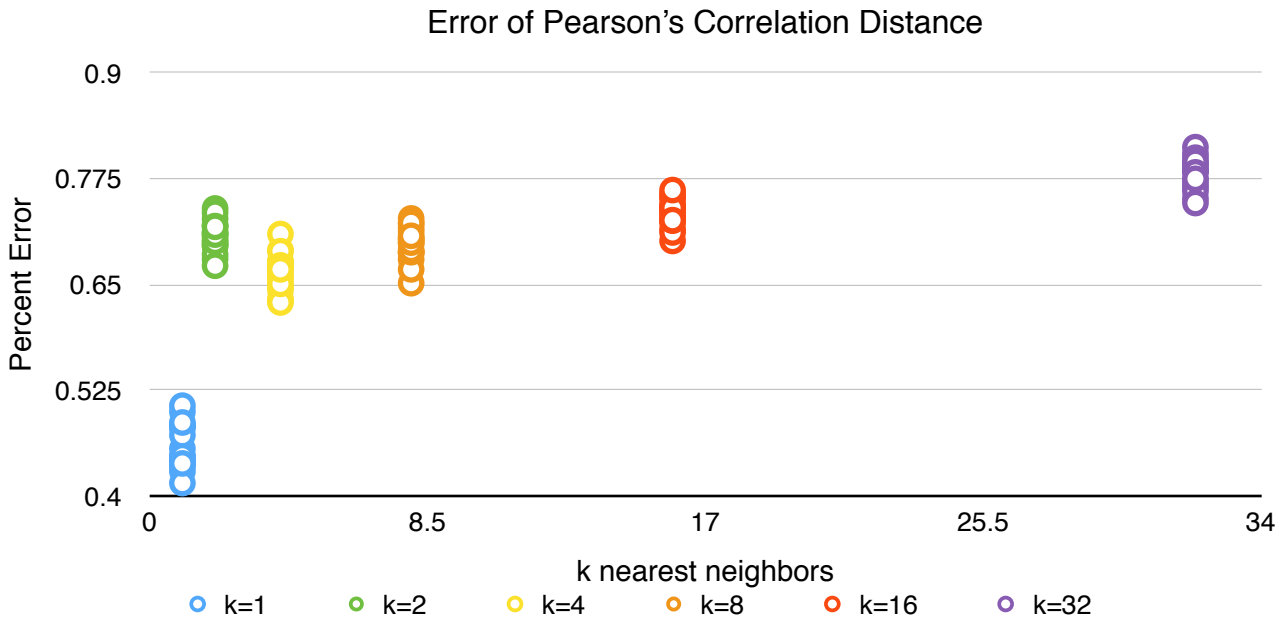


Pearson:
The trend from increasing k had the same affect on the error as with Manhattan distance. It seems as we increased k, the error increased. Having a k of 1 consistently allowed us to achieve an error of slightly more than 45% on average. Despite this, a low k value might not necessarily be great when using Pearson's correlation because this might be as a result of the sparsely populated data and the fact that unrated movies are filled in with zeros.

However, despite my theory that the zeros may be swaying the data, when running the Welch's t test, the difference in k = 1 and k = 2 is significant just as it was with Manhattan distance. The online Welch's t test calculator that I used resulted in a p value less than 0.00001 indicating that these sets are statistically different and it is significant. I would do more testing before choosing k = 1.

k = 1	k = 2	k = 4	k = 8	k = 16	k = 32
0.440233333333	0.707533333333	0.6904	0.651366666667	0.70125	0.751516666667
0.482316666667	0.69625	0.6679	0.727883333333	0.757166666667	0.804583333333
0.480566666667	0.727533333333	0.67735	0.68735	0.71515	0.7641
0.49955	0.720233333333	0.662666666667	0.698066666667	0.74005	0.76785
0.434616666667	0.679916666667	0.63285	0.679566666667	0.7534	0.792233333333
0.48395	0.739633333333	0.70945	0.701016666667	0.7233	0.772816666667
0.482783333333	0.710616666667	0.630533333333	0.704333333333	0.7385	0.7893
0.455933333333	0.7033	0.673733333333	0.72175	0.751433333333	0.812066666667
0.4467	0.694966666667	0.6554	0.7166	0.743766666667	0.784383333333
0.471883333333	0.698883333333	0.659616666667	0.687433333333	0.730883333333	0.759683333333
0.5067	0.734583333333	0.6772	0.6677	0.711266666667	0.760166666667
0.445483333333	0.7118	0.6476	0.687983333333	0.733166666667	0.799633333333
0.438966666667	0.718833333333	0.689166666667	0.704366666667	0.743383333333	0.784216666667
0.429033333333	0.685133333333	0.64295	0.6907	0.725233333333	0.746033333333
0.415266666667	0.671983333333	0.628566666667	0.712433333333	0.738266666667	0.794683333333
0.44695	0.700316666667	0.65055	0.6993	0.740366666667	0.7823
0.486816666667	0.70875	0.672433333333	0.722616666667	0.761383333333	0.78115
0.438466666667	0.71795	0.667683333333	0.707333333333	0.725816666667	0.774666666667

Below is a graph of the error of Pearson's correlation as a function of k. As k increased, error did as well.



Part F

I wasn't able to fully run my item-based variant and compare the filter to the user-based variant with enough data to determine statistical significance between the two. However, it seemed that given the error of the first few folds of k = 1 test for item-based generation, using Pearson's correlation, and i = 0 that item-based generation out performed user based generation. Without more data it is impossible to say if this theory holds; however, based on this small chunk of data, I found that item-based collaborative filtering may be better. The reason this might be the case i that through item based filtering (the way it was posed we write the code for this question) it excludes zero values and only includes numbers with significance for that movie. Thus, item-based filtering would narrow things down better for collaborative filter than user-based filtering.